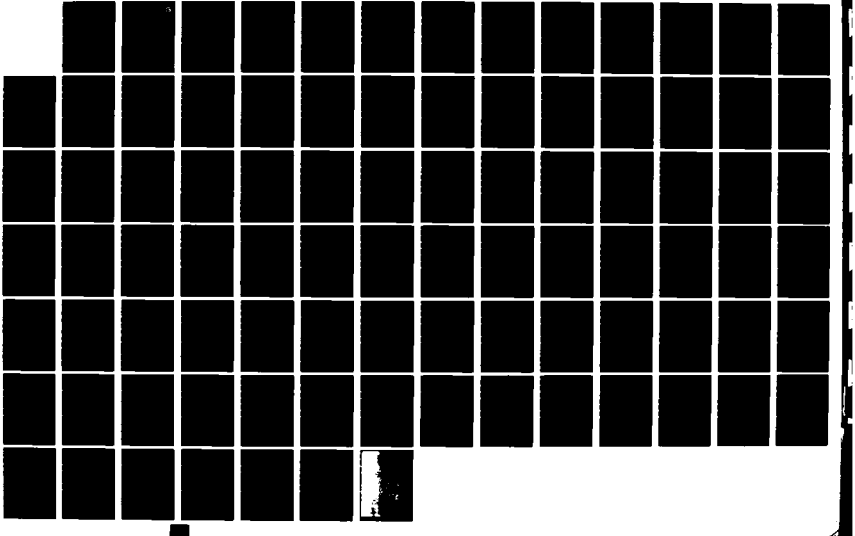
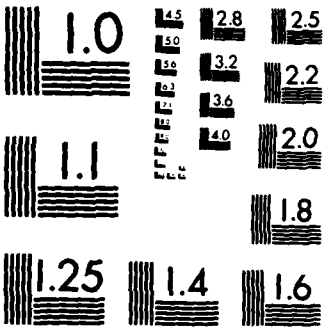
AD-A129 022 NSW (NATIONAL SOFTWARE WORKS) EXECUTIVE ENHANCEMENTS II 1/1
(U) MASSACHUSETTS COMPUTER ASSOCIATES INC WAKEFIELD
C A MUNTZ MAR 83 CADD-8301-3001 RADC-TR-83-59
UNCLASSIFIED F30602-81-C-0208 F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD A129022

RADC-TR-83-59
Final Technical Report
March 1983

NSW EXECUTIVE ENHANCEMENTS II

Massachusetts Computer Associates, Inc.

Charles A. Muntz

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

DTIC FILE COPY

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441

DTIC
JUN 7 1983
A

83 06 07 057

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-83-59 has been reviewed and is approved for publication.

APPROVED: Patricia J. Baskinger

PATRICIA J. BASKINGER
Project Engineer

APPROVED:

JOHN J. MARCINIAK, Colonel, USAF
Chief, Command and Control Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (COTD) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document requires that it be returned.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-83-59	2. GOVT ACCESSION NO. AD-A129 022	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) NSW EXECUTIVE ENHANCEMENTS II		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report 7 May 81 - 30 Sep 82
		6. PERFORMING ORG. REPORT NUMBER CADD - 8301 - 3001
7. AUTHOR(s) Charles A. Muntz		8. CONTRACT OR GRANT NUMBER(s) F30602-81-C-0208
9. PERFORMING ORGANIZATION NAME AND ADDRESS Massachusetts Computer Associates, Inc. 26 Princess Street Wakefield MA 01880		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 63728F 25310116
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (COTD) Griffiss AFB NY 13441		12. REPORT DATE March 1983
		13. NUMBER OF PAGES 94
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Same		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Patricia J. Baskinger (COTD)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer Networks
Software Systems
Network Operating Systems

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The National Software Works (NSW) represents a significant evolutionary step in the fields of distributed processing and network operating systems. Its ambitious goal has been to link the resources of a set of geographically distributed and heterogeneous hosts with an operating system which would appear as a single entity to a user. It is principally aimed at the development of software systems and at providing software tools which can be used to support the software development activity throughout

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

its life cycle. This report describes the current status of the NSW system as well as highlights the enhancements and improvements made to the NSW system during the past two years.

DTIC COPY INSPECTED 2

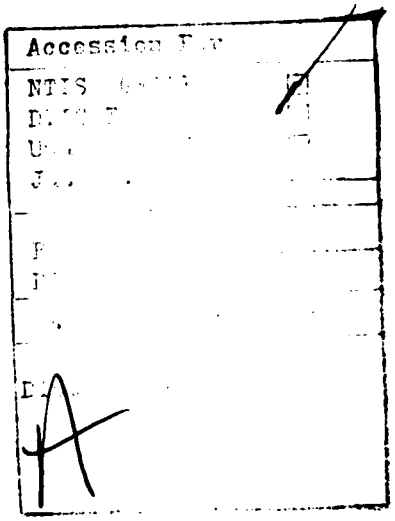


Table of Contents

1. Introduction 1

2. History of the NSW Project 5

2.1. NSW Goals 5
2.1.1. Original Directions 5
2.1.2. Changed Directions 7
2.2. NSW Architecture 9
2.2.1. NSW Components 10
2.2.2. NSW Monitor and File System 14
2.3. Phases of NSW Development 15
2.3.1. Structural Design and Feasibility Demonstration 15
2.3.2. Detailed Component Design 16
2.3.3. Prototype Implementation 17
2.3.4. Reliability and Performance Improvement 18
2.3.5. Production System 20

3. Overview of the Current NSW System 23

4. New Features 27

5. Current Status: NSW Components 31

5.1. Core System Components 31
5.1.1. Works Manager 31
5.1.1.1. Background 31
5.1.1.2. Recent Changes 33
5.1.2. Checkpointer 34
5.1.3. Works Manager Operator 35
5.1.4. Operator Utility 37
5.1.5. Global Configuration Database 37
5.2. TOPS20 Tool Bearing Host Components 38
5.2.1. MSG 38
5.2.2. Foreman 39
5.2.2.1. Background 40
5.2.2.2. Recent Changes 40
5.2.3. File Package 41
5.3. UCLA/IBM Tool Bearing Host Components 43
5.3.1. MSG Central 44
5.3.2. The BJP Package 44
5.3.3. The FOREMAN Package 45
5.3.4. The "FILE PACKAGE" Package 47
5.4. MULTICS Tool Bearing Host Components 49
5.4.1. MSG 49

5.4.2. Foreman 49
5.4.3. File Package 50
5.5. Front End Components 51
5.5.1. UNIX Front End 52
5.5.2. TOPS20 Front End 52

6. Quality Assurance 55

6.1. Introduction 55
6.2. Configuration Management 56
6.3. Software Trouble Reports 60
6.4. Testing 61

7. Future Directions: Conversion to TCP/IP 67

7.1. Background 68
7.2. More about TCP 70
7.2.1. TCP/AHHP Differences 71
7.2.2. AAHP to TCP Transition 72
7.2.3. TCP Support on TOPS-20 and TENEX 72
7.3. More about MSG 73
7.3.1. TENEX vs TOPS-20 MSG 75
7.3.2. TOPS-20 MSG 76
7.4. NSWSTA -- MSG Subtool 77
7.5. Work to be Done 78
7.5.1. MSG-to-MSG Protocol 79
7.5.2. Alternate NSW Configurations 80
7.5.3. 8-bit Connections 81
7.5.4. Host Classification 82
7.5.5. Alarms 82
7.5.6. TOPS-20 MSG Specifics 82
7.5.7. Quality Assurance for MSG 84
7.5.8. NSW Dispatcher 85

CHAPTER 1

INTRODUCTION

The National Software Works (NSW) represents a significant evolutionary step in the fields of distributed processing and network operating systems. Since NSW's inception, its ambitious goal has been to link the resources of a set of geographically distributed and heterogeneous hosts with an operating system which would appear as a single entity to a user. In its present form, NSW provides an excellent medium for evaluating the benefits of such a system.

The National Software Works has been developed in response to a growing concern over the high cost of software. The Air Force has estimated, for example, that by 1985 software expenditures will be over 90% of total computer system costs. In the attack on the cost and complexity of developing and maintaining software, both industry and government have made enormous investments in software tools -- automated aids for the implementors of software and for the managers of software projects. These tools include compilers, editors, debuggers, design systems, test management tools, etc.

As a result of this investment, a large inventory of excellent tools has come into being, so that the difficulty confronting the management of a programming project lies not in the existence of suitable tools, but in

their availability. If some essential tool does not happen to have a version which runs on a computer to which the project has access, the manager is forced to choose among the expensive alternatives of (1) foregoing use of the tool, (2) undertaking to acquire or produce a surrogate tool on his hardware, or (3) purchasing (access to) a computing system on which the tool does run.

Alternative (1) -- not using the tool -- has become prohibitively expensive as the size and complexity of programming projects has grown: most projects can hardly choose not to use a compiler, and this, for instance, may lead to choice of a sub-optimal programming language merely because a compiler for it is available.

Alternative (2) amounts to the software portability problem. The computing field has been chipping away at this intractable problem for years, by encouraging the use of standard -- or at least popular -- programming languages, having compilers on many different computers. Lacking this facility, the project must invest effort at the beginning to re-program -- and debug -- the tools it needs before starting on its actual task.

Alternative (3) has been attacked most effectively with networking. Indeed, it was the marked success of the Arpanet in providing programmers

economical access to diverse computers which provided the foundation on which the concept of a "National Software Works" was built. Instead of moving the software from host to host, let the programmer (and manager) use each software tool on whatever host it already occupies.

We can say that NSW tries, in effect, to make Alternative (3) more attractive, by providing the Arpanet without some of its drawbacks. Using the Arpanet in straightforward fashion, by using Telnet and FTP to access hosts other than one's "home" system, does indeed give you a much wider domain of action, but nonetheless:

- You need an account on each host. This involves the allocation of funding, drawing up contracts, etc.
- The operating system on each host is different, so you must learn different login procedures, command languages, interrupt characters, file naming conventions, etc. Further, you must not confuse each system's conventions as you move from tool to tool.
- Files output from one tool (say QEDX on MULTICS) are to be input to another tool (say CMS2M on IBM 360). This involves at least network transmission and usually file reformatting. To appreciate the magnitude of this problem one should try to use FTP (Arpanet File Transfer Protocol) to move a QEDX output file -- a sequential file of 9-bit ASCII characters in 36-bit words -- to an IBM 360 to be a CMS2M input file -- a blocked file of 80 EBCDIC character records in 32-bit words.

These and similar problems will be familiar to anyone who has used several different systems.

The purpose of NSW is to make this solution (of providing programmers access to tools on different hosts) a practical reality. The NSW user should not have to know about OS/360, TOPS20, and MULTICS with their differing file systems, login procedures, system commands, etc.; knowledge of how to use the individual tools which are needed for the job should suffice. He should not have to worry about reformatting and moving files from a 360 to a TOPS20; file transmission should be completely transparent. The user should not have to worry about obtaining accounts on many different machines, but instead should have a single NSW account.

Thus, the overall design goals of the National Software Works are to provide programmers with a

- Unified tool kit -- distributed over many hosts -- and a
- Single monitor with
    * uniform command language,
    * global file system,
    * single access control, accounting, and auditing mechanism.

CHAPTER 2

HISTORY OF THE NSW PROJECT

2.1. NSW Goals

2.1.1. Original Directions

As originally conceived, NSW was to provide a unified cross-net operating system of the kind described in the Introduction, and with the following specific external goals in mind:

1. The first such goal was large scale.

   Contemporary operating systems support tens of concurrent users; NSW was to support many more users, possibly as many as one thousand. The catalogue alone of the file system for that many users could easily fill a large disk pack. And the table space required for keeping track of a thousand users and the software tools they are using could easily exceed the virtual memory of TOPS20.

2. The second goal was high reliability.

   If there are one thousand online users, then a two-hour system failure costs one man-year of work. The National Software Works -- particularly its monitor and file system -- must degrade gracefully. Failure of a single component -- e.g., a TOPS20 system on which tools are running -- must only reduce system capacity, not destroy it. Further, only those users actually using a failed component should be affected by its failure.

3. The third goal was support of project management.

   NSW was to provide to managers of software projects a collection of programs, called management tools, which they could use to monitor and control project activities. The underlying assumption here is that a manager's ability to insure that each programmer's efforts contribute most effectively to overall

project goals can be greatly enhanced by automating routine management tasks. Furthermore, it is assumed that a good environment for this automation is the system which supports the project programming activities, because it represents an effective point for monitoring and controlling those activities.

4. The fourth goal of NSW was practicality.

   NSW was not to be a "blue sky" system, whose implementation required unrealistic assumptions about its environment. In particular, "practicality" meant:

   - Minimum modifications to existing operating systems on Arpanet hosts. "Minimum" was, in fact, to be construed as "none". It was permissible, if necessary, to add privileged (i.e., non-user) code to existing systems, but the solution to the problem should not depend on rewriting the kernel of any existing operating system.

   - Minimum modifications to existing tools. Here, "minimum" no longer meant "none". It was permissible to require some change to a tool as part of the process of installing it in NSW, but such changes should be small-scale and straightforward.

   - Maximum generality. Any solution which permits the easy installation of existing tools must also allow the easy construction and installation of new tools.

   - No experimental hardware. This requirement meant that new hardware-oriented approaches to reliability -- e.g., PLURIBUS -- could not be used. The NSW monitor and file system are to run on already available Arpanet hosts.

Although it was never stated as explicitly as the goals enumerated above, it seemed to follow from the whole concept of a National Software Works that it would be the sole on-line working environment for most of its users. As we went about the initial design, our mental picture of the user's activity was that he would log into NSW and stay in that environment

for all of his machine activity -- cheerfully using any tool he had access to, without even knowing, or caring, on what host in the network it was running, or where his files were stored.

2.1.2. Changed Directions

As the initial design was completed and the prototype NSW was implemented and used, the original goals had to be somewhat modified.

The original goal of supporting as many as one thousand users with a single NSW proved not to be economically feasible in the present state of the art: it would have required excessive hardware resources, and might still not have shown acceptable responsiveness. Hence, the current plan is to produce a self-contained, moderate-sized system -- supporting up to a hundred users -- which can be distributed in toto; that is, a number of such systems could be in operation independently on the same network.

Given this reduction in scale, the projected implementation of a widely distributed, fully replicated, synchronized data base -- the design of which was well along -- was also seen as an inappropriately large investment. If each NSW system then will have a single monitor host, the principal reliability issue becomes one of preventing a monitor-host crash from aborting ongoing tool operations. Thus, rather than casting the monitor as a set of cooperating distributed processes, as was designed in

the large-scale reliability plan, we allow tool execution to continue despite the absence of the monitor, given that the user's rights have initially been verified by the monitor, and by allowing the results of tool executions to be held indefinitely until the monitor is again available to accept delivery.

Our mental picture of the utilization of the current NSW implementation has also changed. At current hardware and network speeds, there is a human-time overhead to be paid for using NSW, and anyone with experience on a present-day timesharing system will be conscious of it. Use of a single tool is not too seriously degraded, compared to normal TIP-to-host operation, but inter-tool communication by passing files through the central system seems to take a long time when compared with such operations within a single-host operating system. If the user does indeed have to use multiple tools at separate hosts, then using NSW is a pleasure in comparison with going through the scenarios necessary to perform the same operations using Telnet and FTP.

We believe, that there are at present valid uses for such a "Network Operating System" facility, and that there will be an even bigger place for it in the future, considering the forecasts of enormous interconnected networks of personal computers, mass stores, high-performance mainframes, and special-purpose devices.

Our current picture of the "heavy" user of NSW is that he will work, as he does now, primarily within his "home" host system (though he might establish contact with it via NSW mechanisms), and he will use NSW facilities, when he needs to, on an escape basis -- he will signal NSW that he has produced files to be placed in the NSW File System for others to access, or that he needs to execute some tool on a "foreign" host. Later sections on the work of the Analysis Group will discuss the embodiment of this changed perspective.

The design goals as stated in the Introduction, then, might be updated to say that the intent is to provide programmers -- and project managers -- with a wider and more automated environment, including:

- Easy access to services not normally available on their home systems;
- Controlled access to a cross-net file system of text files, programs, and services;
- A structure and some tools for project coordination and configuration management.

2.2. NSW Architecture

In this section, we summarize the overall structure of the NSW, as it evolved to meet the original design goals of a dispersed toolkit accessed through a central monitor.

2.2.1. NSW Components

The requirement is that many users on many different hosts be able to access many tools on many different hosts, under control of a central monitor, and with reference to a global file system. Analyzing this system according to the functions that must be available on each host, we were led to specify a number of functional processes, whose embodiments are called "NSW Components".

By its very definition, NSW is a distributed system. Tool processes run on different Arpanet hosts, and the monitor process must run on at least one Arpanet host; hence, there must be some form of inter-host process-to-process communication. There are low level Arpanet protocols for moving bits from host to host, and there are also several higher level protocols for moving files and for terminal communication. None of these protocols, however, is oriented toward the kind of inter-process communication which NSW requires. Moreover, even though NSW is being implemented on the Arpanet, we want to keep it as independent as possible of the underlying milieu. Network technology is evolving, and we wish to be able to realize the NSW architecture on tomorrow's networks as well. Hence, the first technical problem to be solved is the definition and implementation of an appropriate inter-host inter-process communication protocol. The protocol developed for NSW is called MSG, and the component on each host which implements the protocol is also called MSG.

When the NSW user runs an interactive tool within NSW, either NSW must arrange that his terminal appear to the tool as if it were a simple user terminal on the native host -- this is most appropriate when adapting existing tool programs to run within NSW -- or the tool must be able to perform its input/output transactions via the MSG protocol. As well as talking to tools, the user will be giving commands directly to the NSW monitor, and it is not feasible for all NSW users to have direct terminal access to the monitor host. Therefore, the user will be represented within the NSW system by a process (on any host, in principle) which will communicate for him with the monitor via MSG, and which will handle his connections to tools. This component is called the Front End.

A tool running on some machine makes system calls requesting resources -- primarily file access. Since access to NSW system resources is to be controlled, accounted for, and audited by the NSW monitor, such requests must be diverted from the local system and referred instead to the NSW monitor. In addition, if the tool is interactive, it expects to have a terminal for communication with the user, and this in NSW is via the Front End. So, without modifying the operating system, we must divert the tool's communications with the user and the tool's requests for NSW resources, while still allowing the tool to obtain local-service access of other kinds. The NSW component which solves this problem is called the Foreman.

Batch tools are best described as those whose input and output can be completely specified before tool execution begins. Such tools should not (and often cannot) be supervised from a terminal. Rather, the central monitor works together with a component called the Batch Job Package, running on the same host as the batch tool, to supervise execution of such "absentee" computations.

It was understood from the beginning that there would be no physically separate NSW file-storage media -- NSW files physically reside within the filing systems of the participating hosts, ideally on (or "near") the hosts on which they are most likely to be needed. Hence, there is need for a process on each type of host which understands that host's file system, and can manage the portion of the host system which is available for NSW file storage. Also, we expect that the output of one tool will be used as input to another tool. Unfortunately, if the first tool is a MULTICS editor and the second an IBM 360 compiler, this operation involves character translation (ASCII to EBCDIC), file reformatting (sequential file to blocked record file), and file movement (across the Arpanet). To handle these functions of file storage, transformation, and movement, there is an NSW component called the File Package.

It is worth noting at this point that all of the above components are distributed. Every host in NSW has an MSG server process. Every site to

which a user is connected has a Front End. Every tool-bearing host has a Foreman. Every host on which NSW files are stored has a File Package. It is also worth noting that implementation details of these components vary from host to host. A MULTICS Foreman will be vastly different from an IBM 360 Foreman. Functional specifications for these components are fixed throughout NSW, but implementation and optimization decisions are left free.

And finally, there must be a component for the NSW monitor, or at least for those normal monitor functions which are not already performed by the Foreman (service calls for file access) or the Front End (terminal handling, command interpretation). This component is named the Works Manager; we shall discuss it in more detail in the next subsection.

Let us then summarize the major functional areas of the NSW design and the components which embody those functions:

- Inter-host inter-process communication	MSG
- User interface	Front End
- Diversion of communication with local operating system	Foreman
- Supervision of batch jobs	Batch Job Package
- File storage, transformation, and movement	File Package
- Monitor functions	Works Manager

2.2.2. NSW Monitor and File System

The design of the NSW Monitor -- called the Works Manager -- was probably more affected than any other component by the goals of NSW. Functionally it is not different from any other conventional access-checking, resource-granting monitor (though the mechanisms for defining access domains are unusual). Structurally, however, it is significantly different.

The goals of providing both large scale and reliability on conventional hardware led to a design for distributing the Works Manager and file system. If there are many instances of the NSW monitor on many different hosts, then failure of a host is not catastrophic. Unfortunately, distribution runs counter to the problem-required logical unity of the monitor and file system. If a user inserts a file into the file system using one tool and one instance of the file system, and then requests the same file using a different tool and a different instance of the file system, the two instances of the file system must share a common file catalogue for the system to behave properly. Similarly, all instances of the monitor must share an access-rights data base for proper validation of user requests to run tools.

As previously mentioned, ambitious design has been shelved. A "large" NSW system -- with multiple Works Managers -- is still feasible, but with a

partitioned (rather than a replicated) data base: each Works Manager will control the resources in that piece of the partitioned data base which it "owns", but will have to negotiate with another Works Manager that "owns" other resources. This strategy requires minimum synchronization while providing advantages in reliability and robustness.

2.3. Phases of NSW Development

The design and implementation of the National Software Works has proceeded in five slightly overlapping phases:

1. Structural design and feasibility demonstration
2. Detailed component design
3. Prototype implementation
4. Reliability and performance improvement
5. Management Plan and Production System

In the following subsections we describe these phases in more detail.

2.3.1. Structural Design and Feasibility Demonstration

The first phase of NSW development began in July 1974 and concluded in November 1975. During this period, the basic architecture of NSW (described in Section 2.2) was established. Further, relatively ad hoc implementations of major components were made. These components were integrated into a system which was demonstrated to ARPA and Air Force

personnel at Gunter AFB in November 1975. This demonstration exhibited various system functions, the use of batch tools on the IBM 360 and Burroughs B4700, the use of interactive tools on TENEX, transparent file motion and translation, and a primitive set of project management functions.

This demonstration confirmed that the expected NSW facilities could be implemented and that transparent use of a distributed tool kit was feasible. The NSW System, however, was inefficient and fragile. Further, many of the ad hoc implementations had design weaknesses which limited their general application to a sufficiently broad range of hosts and capabilities. For these reasons, an effort was begun to produce adequate component designs.

2.3.2. Detailed Component Design

This second phase of NSW development was begun in June 1975 with the initial MSG design document. Specifications were developed for Tool-Bearing Host components -- MSG, Foreman, and File Package. All of these specification documents were completed by March 1976. (They have all been revised since then, but the original specifications are still substantially correct.)

During the same period, the external specification of the Works Manager

was also produced. Again, although this specification has subsequently been revised, it is still substantially correct. The remaining portions of the core of NSW -- i.e., the batch tool facility, consisting of the Works Manager Operator, Interactive Batch Specifier, and Interface Protocol -- were designed during phase one, and those designs were retained until phase four.

The remaining major NSW component, the Front End, was the subject of several design efforts. Three incomplete specification documents were produced but none of these was wholly satisfactory. Nevertheless, sufficient design to allow implementation of a functionally correct Front End was accomplished.

2.3.3. Prototype Implementation

As specification documents were completed, various contractors began implementation of the NSW components on the initial set of hosts -- TENEX, MULTICS, and IBM 360; these efforts commenced in January 1976. Implementation on TENEX proceeded more quickly than the efforts on the other hosts -- primarily because the MSG system designers were also TENEX implementors. By October 1976 prototype implementations which conformed to the published specifications had been made for all TENEX TBH components. In addition, all components of the core system were available on TENEX.

Implementation of TBH components on MULTICS and IBM 360 proceeded more slowly; however, initial implementations of MSG components on both of these hosts were completed by the end of 1976. By November 1976 sufficient progress had been made on implementation of a File Package and Foreman on MULTICS that it was possible to demonstrate an interactive tool running on MULTICS. Progress on implementation of 360 (interactive) TBH components reached a similar position in September 1977.

Also during this phase, a TENEX Front End which functionally supported the Works Manager and Foreman according to the appropriate specifications was implemented.

An NSW system containing prototype implementations according to the specifications of the core system, TENEX TBH components, TENEX Front End, batch IBM 360 tools, as well as a rudimentary MULTICS interactive tool was demonstrated to Air Force and ARPA personnel in November 1976. At the same time, a demonstration of MSG components on all three hosts was also given.

2.3.4. Reliability and Performance Improvement

Even though implementation of components on MULTICS and IBM 360 was lagging, implementation of the core system, TENEX TBH components, and TENEX Front End had proceeded to the point that the issues of reliability and performance assumed major importance. The system exhibited sufficient

functional capability that it could clearly support use by programmers if it were sufficiently robust and responsive.

The first task attacked was to provide robustness. Work had begun in 1975 on a full-scale NSW Reliability Plan -- this was the design for multiple Works Managers with duplicate data bases. This detailed Plan was released in January 1977. Since it was clear that implementation of the full Plan was a major undertaking, a less ambitious Interim Reliability Plan which ensured against loss of a user's files was begun in mid-1976. This Plan was also released in January 1977. By June 1977 the core system, TENEX Foreman, and TENEX Front End had been modified to incorporate the features of that Interim Plan. In addition, both the MULTICS and IBM 360 Foremen (only partially implemented) were altered to conform externally to the scenarios specified by the Interim Reliability Plan. A system exhibiting the new scenarios was released for use in June 1977.

Performance of NSW had been slow from the initial implementation. The reasons for slow response were many:

- Interaction between components was by a thin wire (MSG and the Arpanet).
- NSW components (which constitute an operating system) nevertheless were executed as user processes under the local host operating system.
- Component implementation had been oriented towards ease of debugging and other concerns of prototype systems rather than

towards the performance expected of a production system.

In 1977, efforts to improve NSW performance were begun.

The first effort was the development of a performance measuring package for TENEX MSG. Results of the first set of measurements were reported in April 1977; and a number of more sophisticated measuring packages were complete by February 1978. By May 1978, all TENEX components had been instrumented and measurements of page use, CPU time, elapsed time, use of JSYSes (TENEX monitor calls), etc., had been taken under a variety of system load conditions and on several different TENEX hosts. Efforts were then undertaken to make the performance improvements suggested by these measurements.

2.3.5. Production System

Concurrent with the effort to improve NSW reliability and performance, an effort to make NSW a more packaged product were begun. Regression tests for the externally available NSW user system were developed and applied to each system release. A user's manual for the system was published. Documentation of the core system was produced. Finally, a draft configuration management plan was developed.

Work was begun in late 1978 to establish NSW as a software product. The

NSW Management Plan was generated. This document identified a number of roles associated with development, operation, and support of NSW as a product. Briefly these are as follows:

Role	Responsibilities	Organization
(PG) Policy Group	Requirements and Policy	RADC/ARPA
(PDC) Product Development	Product Definition	GSG
(OPS) NSW Operations	Operations; User Support	GSG
(ACC) Architecture Control	Product Integration	COMPASS
(DMC) Development and Maintenance	Component Development and Maintenance	BBN, COMPASS, HIS, UCLA
(TM) Tool Manager	Tool Management	IITRI

A product baseline has been established to bring NSW under Configuration Management. A reasonably complete set of requirements/specification documents has been installed as a set of files in the NSW User System. NSW's Information Retrieval System allows queries on sets of documents; the naming scheme for the baseline demonstrates some of the power of NSW's file naming capabilities:

Document Type	Name Syntax
Requirements	NSW.REQUIREMENTS...
A-level Specifications	NSW.A-SPEC...
B-level	NSW.B-SPEC.<component>...

Specifications
for <component>

C-level NSW.C-SPEC.<operating-system>.<component>...
Specifications
for <component>
on <operating-system>

where the variables take on values as follows:

<component>	<operating-system>
BJP	TENEX
FE	OS360
FL	MULTICS
FM	UNIX
FP	...
...	

The revision level of all of these documents is also noted, as part of the file name.

In an effort to "productize" NSW, the current version of NSW has a number of new features that makes NSW more user friendly and reliable; these new features are more fully described later on.

A computerized tool for coordinating the reporting, checking, fixing, and testing of software problems has been developed and put into use: it is called MONSTR -- a MONitor for Software Trouble Reporting. It is driven by tabled protocols defining the desired interactions between all the organizations listed above, and handles the passage of messages through the appropriate channels between them.

CHAPTER 3

OVERVIEW OF THE CURRENT NSW SYSTEM

The NSW system currently available to users is NSW 6.0, first released in January 1982. It offers the following general features:

- Twenty-one interactive TOPS20 tools, running under a new Foreman that contains an improved WM-to-FM database synchronization scheme and contains the new Work-Space Command Interpreter.
- Twelve interactive MULTICS tools, running with the improved WM-to-FM database synchronization scheme.
- Nine interactive IBM 360 tools, and fifteen IBM 360 batch tools. During this contract period NSW components had to be adapted to the new IBM MVS operating system.
- Core components (Works Manager and Works Manager Operator) and a TOPS20 Front End, interpreting and implementing a the set of system commands.
- A totally revised NSW User's Reference Manual.
- The normal configuration of NSW 6.0 includes the following hosts:
  - USC-ECLC (TOPS20 — Works Manager, Tools)
  - RADC-TOPS20 (Tool-Bearing Host)
  - CCN-360/91 (Batch and Interactive tools)
  - RADC-MULTICS (Tool-Bearing Host)
- The release procedure is formalized and partially automated.
  * A source code repository is maintained for the life of a release.
  * A semi-automatic configuration control facility is implemented, to handle site-specific parameters.
  * A release-specific document is produced for each new

release, detailing the configuration and installation procedures, operational procedures, and changes from the previous release.

- The MONSTR tool for cataloging and tracking Software Trouble reports, and their associated fixes, has been used.
- Both the UNIX and TOPS20 Front End have extensively been made more user friendly.

The current status of the individual component implementations is presented in the "Current Status: NSW Components" chapter, by host system for the Tool-Bearing Host components, and by functional class for the Core system components and the Front Ends.

During the past contract period, a group of senior NSW personnel, under the chairmanship of Dr. Robert Thomas of Bolt Beranek and Newman, was formed to reconsider the entire NSW design, and to draft a re-design of the entire system. This group was named the "NSW Analysis Group", and its charter was to rethink the organization and functional distribution of the NSW -- not to throw away all the previous design and plan an entirely new Network Operating System, but to reconsider all the design decisions which had been made since 1974, in the light of experience and the "changed directions" concept of the potential uses of NSW.

The group met frequently during 1980, and exchanged working papers. The final report of this effort is entitled "NSW Functional Specification",

Revised Draft September 1980. The conclusions in this document have largely determined the direction of planned changes in NSW.

Following the lead of this effort, a task-list was formulated for the changes and improvements to be made to NSW for the next release, choosing from the features of the "Functional Specification" those which could be applied to the previous Release 5.0, and those which would be of greatest value for the ongoing Air Force Technology Demonstration.

Thus, NSW release 6.0 is a result of the changes and improvements named in the task-list.

CHAPTER 4

NEW FEATURES

The following list itemizes many of the new features of the present NSW 6.0 release. In the course of implementing many of these features many bugs were found and fixed.

1. Revised and Integrated NSW Resource Catalog Design and Implementation

    a. improved and consistent naming, lookup and entry functions

    b. revisions to support single integrated object space, including full support for file and service type objects

    c. revisions to the definition of semaphores

    d. revisions to the scoping mechanism

    e. addition of own space automatically created with new nodes

2. Decentralized Protocols for NSW File Movement and Service Activation

    a. WM procedures for file/service entry, lookup and access control

    b. FM procedures for directly participating in file movement

    c. Protocol Modifications to Avoid Timeout on Long Operations

3. Automatic Cleanup of Old Session on Re-Login Attempt

4. Limited user I/O commands

    a. TYPE command to view text file on user's terminal.

b. LIST command to send that file to the line printer.

5. Extended Services and Service Session Support

a. standard WS command interpreter for the TOPS-20 TBH

b. native mode services for the TOPS-20 TBH

c. limited single host tool kits and chained tools for the TOPS-20 TBH

d. revisions to support direct file access

e. detachable service sessions for the TOPS-20 TBH

f. additional parameter collected and passed to the TBH on service initiation

g. character string arguments collected as part of "USE" command line

h. session id, user login name, service name, etc., available to TBH service

6. Status Commands for the User

a. ability to view static descriptors of file and service system objects, node and session records

b. list of logged-in users

c. status of configuration hosts

d. dynamic status of a user's active services and long transactions

7. TBH/WM Data Base Synchronization Improvements

a. WM comes up -> TBH FM (from config.bas)

8. Other System Changes/Bug Fixes

a. MSG large host-address mode

b. avoiding user password recording in event logging

c. TOPS-20 FM using GFT parameters

d. upgrade WM-BATCH-ENDJOB to propagate accounting list

e. All the TOPS20 components that are written in BCPL were re-compiled with the new BCPL compiler. The code produced by this new compiler is faster and smaller.

f. Additional compiler FE status Queries

CHAPTER 5

CURRENT STATUS: NSW COMPONENTS

5.1. Core System Components

5.1.1. Works Manager

5.1.1.1. Background

At present, the Works Manager consists of a number of identical concurrent instances of the same program, each one working on a single request at a time. All such processes share two common data bases, the Works Manager Table data base and the NSW File Catalogue. In addition to these processes there is a separate process, the Checkpointer, which makes periodic backup copies of the data bases.

The Works Manager supports 41 different Works Manager procedure calls, which are available to other NSW processes. These procedures are described in the Works Manager System/Subsystem Specification and the Works Manager Program Maintenance Manual.

The management/node manipulation tools are implemented entirely within the Works Manager. These are invoked under the tool rights mechanism using the same interactive/HELP technique as batch tools. This has allowed removal of the specialized Front End/Works Manager interface formerly used,

eliminating five special Works Manager procedure calls, and eliminating all knowledge of these tools from the Front End. Thus the UNIX Front End development is also freed of this knowledge.

A file attribute mechanism is also implemented, allowing Foreman calls to the Works Manager for getting and delivering user files to specify the file type. This feature is required to support 360 interactive tool installation.

The Works Manager also uses the Global Configuration File, which has given it more operational flexibility. In particular, its timeouts on calls to remote procedures can be tuned without affecting other components. Also, event logging can be more flexibly specified, and better accommodates the divergent needs of the developers and operators.

The Works Manager, which consists of approximately 31.2K lines of BCPL code, is structured into a number of layers. At the top level, WMMAIN waits for a procedure call message from another NSW process, does initial decoding and validity checking of any such message, then dispatches the message to the proper routine. The Works Manager Routines, WMRTNS, implement the 41 Works Manager Procedures. At their disposal are a number of lower-level utility packages and subsystems. The Works Manager Table Package, WMTPKG, handles all interactions with Works Manager tables. It

serves as an interface to the Information Retrieval System, INFRTV, which manages the NSW File Catalogue and the Works Manager Tables. All NSW processes written in BCPL have available NSUPKG and BCPPKG. NSUPKG contains a number of facilities to handle MSG messages, create and record NSW fault descriptions, etc. BCPPKG provides basic utilities to handle character strings, do searching and sorting, and so forth.

5.1.1.2. Recent Changes

The following list of features covers most of the recent changes included in NSW release 6.0.

- Modified the LOGIN scenario to allow the user to terminate a previously crashed tool-session, if he finds his node "busy" when he attempts to log in.

- Modified the scoping and access-checking rules, according to the conclusions of the Redesign Effort. The result of these changes are described in the "User's Reference Manual".

- File groups - Extended the Works Manager command language so that whole groups of files can be copied, renamed, deleted, etc.

- Full object attributes - In the past only the filename portion of the complete NSW filename could be used for retrieval. Also, the use of file attributes by services were only permitted for the Global File Descriptor. The implementation of object attributes has been completed. Note, now files are a subset of objects.

- System status commands - A routine is now available that shows who is using the system. Also, the status of NSW components is now accessable.

- File space maintenance and management - Automatic file space maintenance (e.g., reconciliation of the Works Manager's File Catalog with directories distributed at TBHs) has been implemented.

- Size and performance improvement -- as the result of using the new BCPL compiler.

5.1.2. Checkpointer

The Checkpointer status mimics that of the Works Manager, since it consists largely of the entire Works Manager utility package, with a relatively small upper layer of code to implement the specific Checkpointer procedures. Performance and size improvements have been realized by recompilation with the new BCPL compiler.

The Checkpointer has the following characteristics:

- Implements the FM-GUARANTEE call on the Foreman required by the Interim Reliability Scenarios.
- Manages NSW file deletion. Files deleted by the user are actually deleted by the Checkpointer after a time interval, as required by the Interim Reliability Plan.
- Makes Checkpoint files of all Works Manager database files at configuration controlled intervals.
- Is robust and flexible to about the same level as the Works Manager itself.
- The Checkpointer received a major re-write for NSW 4.0, and except for one small bug-fix has remained unchanged since then. A completely new asynchronous remote procedure call handler was written. This allows the Checkpointer to make multiple simultaneous remote procedure calls, usually to delete files without interfering with the timing of database checkpoints.
- The Checkpointer also uses the Global Configuration Database file, and has gained significant flexibility as a result. The external procedure call timeout, checkpoint interval, and waiting period before file deletion occurs are all under operator

control.

- The Checkpointer is halted by an interrupt from the operator utility OPRUTL.

5.1.3. Works Manager Operator

The Works Manager Operator has been extensively used with the 360 Batch Job Package since November 1978. Detail improvements have been made to improve reliability and job status reporting.

WMO shares a data base (the Job Queue File) with the Interacive Batch Specifier (IBS) module in the Works Manager. We had intended to remove this shared access by making all access to this data base be via procedure calls on WMO, which will have sole access. To this end, direct access to the data base by the WM to get a batch job status (NSW: JOB) has been replaced by a call on WMO by WM on the WMO-SHOWJOB procedure. Direct access to the data base by IBS may be replaced by use of a WMO procedure, WMO-ENTERJOB, to be specified and implemented in the future.

WMO also uses the configuration database file.

Some notable characteristics of the current WMO are as follows:

- WMO is responsible for both processing the Job Queue File and handling WMO procedure calls. These two tasks are handled by distinct instances of WMO in any given NSW system.

1. There is exactly one instance of WMO processing the job queues. A standard locking discipline guarantees that precisely one such instance exists. This instance executes the job steps necessary to process a batch job, and initiates all procedure calls to external processes (WM, BJP, FP). It never receives generically addressed MSG messages.

2. There are zero or more instances of WMO which receive generically addressed MSG messages, and handle all currently defined WMO procedures. These instances never execute job steps or initiate external procedure calls. Thus, these instance(s) provide external access to the data base.

- A primitive retry mechanism exists. WMO will retry an external procedure call indefinitely when it fails due to network or remote host crash. It will retry a failed external procedure call a maximum of three times if the failure is due to resource problems, e.g. no disk space.

- Status reports generated by WMO for display by WM (NSW: JOB) report all information supplied by BJP.

- The maximum number of jobs in the job queue file is currently 64. This may be increased when needed, but requires re-compilation and reloading of WMO.

- The WMO cycle number may be set manually by the WMO utility (WMOUTL), but does not automatically increment with each cold start. "Cold Start" in this version occurs only when a new job queue file is created.

The Works Manager Operator program has proven to be quite reliable in the face of Works Manager, network, and batch-host failure; no significant changes have had to be made for some time.

5.1.4. Operator Utility

The operator utility program, OPRUTL, has become sophisticated enough to be mentioned as a core system component in its own right. It can operate either stand-alone or as an actual NSW component under MSG. It allows the NSW to perform some maintenance functions which are tedious with more primitive developer-oriented utilities. Its capabilities are:

- To clean all or specified LOGIN entries out of the Works Manager database, e.g., to recover from a core system host crash.
- To enter new tool descriptors into the Works Manager database. This is the only practical means of entering batch tools, which must be parsed and error-checked on entry.
- To stop the Checkpointer via an MSG alarm.
- To report on current NSW usage, i.e., who is logged in.
- To reset all internal database locks (as a cleanup operation).
- To operate the Fault Logger, displaying selected portions of the logs on file.

5.1.5. Global Configuration Database

NSW 4.0 included initial use of a prototype configuration database which supports specification and control of site dependent configuration data; no further development of this usage has proved necessary since that time. It consists of a specially formatted text file containing parameters which apply both to a whole NSW configuration - hosts used, MSG generic classes defined, etc - and to a specific host in the configuration - directories

used, timeout values, logging parameters, etc. The database is designed to be maintained at a central site and then be broadcast to all hosts in a given configuration, giving NSW operations a centralized means of controlling an entire NSW configuration.

The current database is a prototype used by most of the core system components and the TENEX/TOPS20 File Package.

Configuration data now used include:

- WM - system herald, remote call timeout, event logging, public scopes and public keys.
- CHKPTR - remote call timeout, deleted file wait interval, checkpoint interval, event logging.
- WMO - remote call timeout, event logging.
- FLPKG - remote call timeout, event logging, filespace directory name.
- FOREMAN -- list of FOREMAN hosts for Works Manager/Tool Bearing Hosts Database Synchronization.

5.2. TOPS20 Tool Bearing Host Components

5.2.1. MSG

The MSG specification was produced in January 1976. It was revised in December 1976 - primarily to resolve ambiguities in the earlier document. It was extended in April 1978 to allow for support of multiple, concurrent

NSW systems. The TOPS20 MSG component implements the revised and extended specification with only two exceptions (which are noted below).

The TOPS20 implementation of MSG is a single executable module which will run under (TENEX, TOPS20 Version 101B, and) TOPS20 Releases 3A, 4 and 5. In addition to the communication functions supported for processes (and defined by the MSG-process interface specification) the TOPS20 implementation includes a powerful process monitoring and debugging facility, and comprehensive performance monitoring software.

The TOPS20 implementation does not perform MSG-MSG authentication. Message sequencing and stream marking are not implemented (however the underlying software structure exists to support both).

MSG supports rapid timeout of attempts to contact remote hosts which are down or where no central MSG is running; this markedly reduces the wait time imposed on a user who has attempted to use an unavailable resource.

5.2.2. Foreman

5.2.2.1. Background

The current TOPS20 Foreman implements all scenario functions defined by the Interim NSW Reliability Plan in its most recent revision.

The TOPS20 Foreman had been extensibly modified as a result of the the extensive performance measurements made in early 1978 and reported in BBN report No. 3847, "A Performance Investigation of the National Software Works System". Performance enhancement had been limited to reducing resource consumption by the Foreman, e.g. by minimizing use of expensive JSYSes, pre-allocating workspace directories, etc.

Since NSW 4.0, the Foreman has incorporated improved reporting of user file delivery from the tool. The Foreman also reports faults to the Fault Logger.

5.2.2.2. Recent Changes

The enhancements made in the last contract period have had perhaps a greater impact on the TOPS20 Foreman than any other NSW component, perhaps excepting the Works Manager itself. The following list contains the most important of these:

- Coordinated Works Manager/Foreman protocol design and implementation to have common data base items reflect local resource management decisions (i.e. WM/TBH database synchronization).
- Incorporated some of the File Package's functionality in order to

optimize file fetching and delivery operations.

- Provided a Work-Space Command Interperter; this allows the output of one tool to be used as the input of another on the same host without having to place these scratch files into NSW object space.

- Allows the execution of tools in un-encapsulated mode. However, no direct calls on NSW are allowed by the tools running in un-encapsulated mode. Thus, the mode allows tools to be run in native mode only.

5.2.3. File Package

The TOPS20 File Package is now functionally complete. The task of writing Intermediate Language encode/decode for non-TENEX binary format files is complete, and has been tested with the CCN/360 File Package for several representative binary file types. The current File Package version has the following characteristics:

- All specified File Package procedures are implemented and tested for local, family, and non-family network transfers. Unspecified procedures to support the obsolete IP mechanisms in WMO had been expunged.

- Avoiding timeout on long file tranfers through the use of ISR (Intermediate Status Reply) Mechanism. Basically, periodically status probes are send out to the FLPKGs involved in the transmission to obtain the status of the transmission; appropiate action is performed when appropiate. Normally, the reply is an "OK" and nothing is done.

- In the past, the Intermediate Language (IL) encode/decode package had been re-structured for greater efficiency and maintainability. Encode/decode had been partitioned into three classes - text files, sequenced text files, and binary files; there is an encode and a decode module for each class, totalling six. Code size had increased, but both efficiency and code comprehensibility have been greatly enhanced. The interface

between the (BCPL) calling routines and the (MACRO10/20) service routines has been simplified. Implementation of binary file encode/decode is complete, and has been extensively tested both against itself (i.e. against a remote TOPS20 simulating a non-TOPS20 host), and against the CCN/360 File Package. We have confirmed correct transmission of CMS2M object files from CCN/360 to TOPS20.

- Performance enhancements have been implemented based on the results of BBN's performance investigation as reported in BBN report No. 3847, "A Performance Investigation of the National Software Works System", Draft Version, July 1978 by Richard E. Schantz. We have minimized the use of expensive JSYSes, notably the CNDIR (connect to directory) JSYS (average cost 220 ms per call). We have done so by specifying that the File Package must be able to create/read/delete files in its own filespace and Foreman workspaces without connecting to them, and letting it stay connected to its LOGIN directory. This has had no practical effect on the operation of NSW, beyond requiring that these directories be accessible from the system LOGIN directory. These enhancements have resulted in a CPU usage reduction of up to 60% for delivery of a file from the Foreman workspace.

- The logging of messages sent/received via MSG is under control of a spec in the Configuration Database (as in WM, WMO and CHKPTR). When logging is disabled, CPU usage for typical FP calls is reduced 25% - 40%. For comparison, the FP retrieval calls analyzed in BBN report No. 3847, "A Performance Investigation of the National Software Works System", Draft Version, July 1978, by Richard E. Schantz, which averaged about 2.9 seconds, can be reduced to as low as 0.7 seconds with logging disabled.

- The File Package is written primarily in BCPL (approximately 6.9K statements including utilities.) The IL encode/decode package is written in Macro-10 and consists of approximately 1.7K instructions.

5.3. UCLA/IBM Tool Bearing Host Components

All the IBM/NSW components had to be modified because of the new IBM MVS system at UCLA. The functionality which was present in the 360/91 MVT version of the NSW system has been restored in the IBM MVS environment.

In summary, for the individual components, this amounted to:

- The BJP had to be redesigned and written from scratch.
- The Foreman had to be heavily modified.
- The File Package needed a modest conversion.
- MSG needed conversion to use IBM virtual terminals instead of the UCLA-specific ones in MVT.
- All subroutine packages needed minor revisions.
- The tools all needed to be re-installed, this time using better installation procedures, in coordination with the Tool Manager contractor.

The following sections summarize the status of the UCLA IBM NSW system by each component package.

Also, during this contract period the Interactive GIM database tool has been installed.

5.3.1. MSG Central

MSG central is a set of routines that execute as a part of the UCLA ARPANET Network Control Program (NCP). It contains all the machinery necessary to create and destroy jobs, to execute the processes that are required to service generic message requests, as well as that needed to switchboard messages between executing processes on the local and remote hosts.

MSG central is written in Assembler language, using a set of subsupervisor interface macros that are specific to the UCLA NCP. It is not directly executable outside this context; however, the UCLA NCP has been successfully exported, and MSG could be exported concurrently.

MSG communicates with actual NSW processes via the UCLA interprocess communication mechanism known as the Exchange. Exchange is an exportable package. The other end of this communications link is managed by subroutine packages, so its elementary characteristics are not known to the using process.

5.3.2. The BJP Package

The BJP package consists of the modules that perform the Batch Job Processor function. It allocates and frees temporary workspaces as batch jobs are created and finalized by the Works Manager Operator (WMO) core

component. It submits local control-language data sets provided by WMO, monitors the submitted jobs, responds to status queries, and notifies WMO ansynchronously when a job is completed.

The BJP is implemented on the UCLA system as a continuously available single MSG process executing as a swapped TSO job. The single process is initiated by MSG central whenever it is itself initialized. The process name "BJP" carries the UCLA local "queue-generic-messages" attribute, which causes a single process to service all incoming generic requests directed to that name.

5.3.3. The FOREMAN Package

The *Foreman* package consists of routines that implement the NSW Interactive Foreman function. It is executed on the UCLA system as a swapped TSO job, of which instances are materialized by MSG central in response to incoming generic messages.

Basically, the Foreman implements a "begintool" transaction which initiates a user-requested tool session. This transaction carries a "program name" which directs the Foreman to a special program stored in a local data base. This program is written in a locally defined language to be interpreted by the "Encapsulator Command Interpretor" (ECI) subcomponent of the Foreman. The Foreman calls the ECI to execute this program, which

defines the file setup and cleanup and the "personality" of the tool. At indicated points in this execution, the ECI returns control to the Foreman to attach an indicated program, typically the native-mode tool program itself, as a concurrent process. On completion of that program, the ECI resumes interpretation of its program.

The Foreman also responds to "endtool" transactions. Because the tool and the Foreman are concurrent processes, the Foreman remains receptive to such transactions from the NSW core system during tool execution; however, due to the blocking mechanisms of the MVS operating system, both the Foreman and the tool are blocked when the tool is waiting for keyboard input from the user. This causes operational problems, and makes it inadvisable to terminate a UCLA-mounted tool by any mechanism except the tool's own voluntary termination command.

Only encapsulated tools are supported, technically, and then only through pre- and post-processing by the ECI. The Foreman does not actually monitor tool execution, and no Foreman/Tool interface for "new" tools is presently defined.

Due to critical main-storage constraints in MVS/TSO, the Foreman can not maintain an LND, so recovery across crashes is not supported. In keeping with this, the "savelnd" and "rebeginservice" procedure calls to the

Foreman are not properly supported. They are accepted, but they function more or less like the "endservice" and "beginservice" requests.

5.3.4. The "FILE PACKAGE" Package

The "File Package" package consists of routines that implement the NSW File Package function. It is implemented on the UCLA system as a swapped TSO job, of which instances are materialized by MSG central in response to incoming generic messages. Unlike Foreman processes, File Package processes are reusable. That is, when an instance of a File Package is complete, the same program instance will rematerialize as a new NSW process which can be used to satisfy another incoming generic message.

The File Package responds to five basic procedure calls:

1. The "import" procedure: moving or copying data into local NSW filespace from an external directory, another host, or the temporary workspace assigned to an NSW batch or interactive tool.
2. The "export" procedure: copying data from local NSW filespace to an external directory or a tool workspace.
3. The "send" procedure: copying data from any local data set to another host.
4. The "transport" procedure: copying data from an external directory or another host into another local external directory.
5. The "delete" procedure: deleting a file copy from any local filespace.

In servicing any of these procedure calls, the File Package performs operations that consist of combinations of several almost-independent capabilities:

- Making equivalent copies of files.
- Translating files from one local representaton ("file type") to another.
- Moving file data between hosts.
- Encoding and decoding the standard NSW Intermediate Language (IL).
- Deleting copies of files.

Due to restrictions in the MVS data security system, the File Package (as well as any other NSW component) can only access data sets stored under its own charge number. Passwords cannot be used to override this constraint.

ASCII-type format effectors are not supported in other than internal forms. The tab descriptor of the GFD is lost when the GFD is relayed to a foreign host in a SENDME call.

The FLPKG supports the ISR (i.e. Intermdiate Status Reply) mechanism; this prevents timeouts on long file transfers.

5.4. MULTICS Tool Bearing Host Components

During this contract period, the Multics components have been upgraded to use the new Works Manager-to-Foreman database synchronization protocol, as well as to use the new NSW Resource Catalogue.

5.4.1. MSG

Previously, the Multics MSG server depended on an ad hoc, unsupported "Tasking Software" extension to the operating system; that Tasking Software is now supported by Multics TBH Support and has already been upgraded.

MSG has been modified for support of extended-leader host addresses, as required by current Arpanet protocols.

Many MSG bugs have been fixed.

5.4.2. Foreman

The original version of the Multics Foreman was implemented to support tools which were written (or rewritten) specifically for use within NSW; the QEDX-RM tool, for instance, is a specially tailored version of a standard Multics editor containing explicit calls on Foreman functions where needed. As the emphasis in NSW has shifted more in the direction of packaging existing tools for easy cross-net access through NSW, an encapsulation technique has been developed and implemented for the Multics

Foreman, permitting a larger number of tools to be made rapidly available.

The robustness of the Foreman has been increased, especially in the areas of detecting and reestablishing broken direct connections to the Front End.

Previously, the only safe non-ABORT method of terminating a tool session was to use the tool's own internal termination command; but now, the Foreman is able to respond to unexpected specific messages, allowing the QUIT TERMINATE and FM-LND-SAVE scenarios to work properly.

Multics tools follow an approach to the file system which differs from that of its TOPS-20 counterpart in that a file write operation is considered to be a global one, not just to a local workspace copy. Therefore, in the event of a crash, the highest version local copy will be equivalent to the global copy. This accomplishes two things:

1. Read and write operations appear to the user to be NSW global operations with the workspace copy as transparent as possible.
2. LND-Saving is not necessary since file delivery is done as needed, not at the end of a tool session.

5.4.3. File Package

The Multics File Package is a fairly reliable component. It conforms closely to the specification, and supports file encodement into Intermediate Language about as well as the other TBH File Packages. The

Intermediate Status Reply mechanism for the prevention of timeouts on long file transfers has also been implemented.

5.5. Front End Components

There exist two operational Front End components for the present NSW System: The UNIX Front End, produced by Bolt Beranek and Newman, and the TOPS20 Front End (the "COMPASS FE"), produced by Massachusetts Computer Associates. There has also existed the "SRI FE", Produced by SRI International, which is no longer maintained, although it did successfully work with earlier versions of the NSW.

Both Front Ends had to be extensivley changed to access (1) the new system status commands, (2) the new catalogue naming scheme and (3) the new FE-to-FM protocols.

Also, both FEs are fully described in the new "NSW User's Reference Manual".

In summary, the following features were added to both FEs:

- A number of new commands, or additions of arguments to current commands, which involved no change in FE programming other than the recognition and translation of the commands themselves:

    * Additional arguments to the USE (service) command (host, workspace, arguments of call to the service invocation,

etc.).

  * System status-information commands (show descriptors, current users, host configuration and status, etc.).

- A command for "detaching" from a Foreman session at the user's request, that is, voluntarily causing the "saved session" activity which now occurs only when the user's connection to the FE is lost.

- Additional communication conventions between the FE and Foreman, covering user signals for talking to the Foreman rather than the tool, and for notifying the user that some output is waiting for him to see from a tool he is not actively communicating with.

- "Local" commands (not forwarded to the WM) for allowing the user to set his terminal-type and other parameters to improve FE handling of the user-interface communication.

5.5.1. UNIX Front End

The new TYPE and LIST commands have been added along with the new keyboard macro facility. Many bugs have been found and fixed.

5.5.2. TOPS20 Front End

As well as syntatically looking more like the UNIX FE, the new TOPS20 FE now knows about all the terminal types that the TOPS20 host knows about.

The major differences between this FE and the UNIX one is (1) no TYPE or LIST command, (2) no keyboard macros, and (3) only front-end return-mode deferred is supported.

Complete recompilation of the FE program with the improved BCPL compiler has improved the efficiency of the running code, and permitted a number of localized improvements to be made.

CHAPTER 6

QUALITY ASSURANCE

6.1. Introduction

The objective of Quality Assurance is to provide continual, reliable access to a product whose limitations are both known and removable over time. It is the assurance that whenever a user accesses the service, it has at least the quality of the product previously accessed, and that notification of any trouble which might arise can be dispatched to a chain of organizations which can address it in a timely and effective manner. Attainment of these objectives rests upon Configuration Management (CM) - to ensure that the approved system revision is currently operational - and Software Trouble Reports (STR) - to respond to and coordinate the removal of limitations.

In this section, we discuss Quality Assurance methods for the NSW as a whole. Most CM is performed as a supporting service by Data Technicians, but response to and removal of limitations requires coordinated participation of many persons: users, operations, management, and developers.

6.2. Configuration Management

CM is to warrant that at all times, the service being offered is that approved by the Policy Group. As NSW is a distributed group of heterogeneous systems, an auditing style of CM is employed. That is, periodic inspections of the names and revision level of service components are compared with the system "inventory" as of its official release. Such auditing should ordinarily produce no exceptions and is best viewed as a watchdog who seldom "barks". Should changes appear, management can trace them to reduced limitations, new features, etc. The day-to-day operation can be delegated to the specialist organization; management personnel need only perform periodic audits.

The "inventory" in the NSW context is really the configuration index of a distributed, heterogeneous system. We decided to produce on each host a local configuration index, and then to copy those indices to a single host in order to construct a system configuration index. Each operating system has a tool suitable for constructing a local configuration index with some useful properties: text files, each line of which identifies a file in a given context (directory, library, etc.), by name and by the time of creation or most recent modification. TOPS-20, TSO, and MULTICS offer DIRECTORY, PDS, and library-map, respectively. Procedurally, the Data Technician can index an NSW system by successively logging into each component host, running its local index-producing tool, copying the result

to a single site, and finally composing the system configuration index by concatenating the local indices.

Every time an approved change is made, an index is prepared and retained. This approved index is used as a comparand in subsequent configuration audits. The periodic audits are performed by constructing the (current) configuration index and using TOPS-20's FILCOM tool for comparison. This is the motivation for single lines of text for each item described: if differences are found, FILCOM will display them in a fashion that is directly usable in pursuing the reason for the change. An example index and change notice are shown below:

```
Configuration Index for host USC-ISIC as WMH
     Configuration files (C. Muntz)
 MSG-GENERIC-NAMES.;103 585(7)  4-Jun-80 05:19:13
 MSG-NETWORK-CONFIGURATION.;102 73(7)  6-May-80 08:14:30
 CONFIG.BAS;514 1001(7)    10-Sep-80 02:40:13
 FORCOMFILE.BASE;3 512(0)  12-Mar-80 11:55:06
     Msg (R. Thomas)
 MSG.SAV;108161 48640(36)  10-Sep-80 02:33:46
     Front End (K. Sattley)
 FE .SAV;65201  33280(36)   8-Sep-80 05:11:33
 FETHDL.EXE;66300 36864(36)  8-Sep-80 05:23:25
 UNTLNT.EXE;3202 19456(36) 30-Jul-80 06:07:38
```

```
     Dispatcher (R. Schantz)
DSPCHR.SAV;1310 6656(36)  17-Apr-79 13:55:22
NSWROOT.SAV;2310 6144(36) 17-Apr-79 19:07:55
     File Package (C. Muntz)
FLPKG.SAV;25700 46592(36) 22-May-80 05:05:47
LOGUTL.SAV;4000 19456(36) 11-Mar-80 07:11:11
     Foreman (M. Marcus)
FOREMAN.SAV;1613 26112(36) 18-Jul-80 13:11:00
MKCOM.SAV;1611 6144(36)   18-Jul-80 13:25:20
LOGRED.SAV;1509 4608(36)  13-Aug-79 09:38:18
     Works Manager (C. Muntz)
WM .SAV;29602  120832(36) 17-Aug-80 15:18:17
CHKPTR.SAV;3401 86528(36) 20-Aug-80 14:47:07
WMO.SAV;16900  33792(36)  11-Mar-80 06:59:01
OPRUTL.SAV;1640 83456(36) 23-Jun-80 04:48:28
WMOUTL.SAV;14700 30208(36) 11-Mar-80 06:58:07
SIMWMT.SAV;5800 85504(36) 25-Jun-80 10:53:27
DMPUTL.SAV;1100 64000(36) 11-Mar-80 07:23:11
SIMINF.SAV;7300 58368(36) 25-Jun-80 10:55:01
DBSTAT.SAV;600 65536(36)  11-Mar-80 07:24:07
SIMWTF.SAV;1400 64512(36) 11-Mar-80 07:19:26
     Fault Logger (B. Shipman)
FL .SAV;2010   36635(36)  18-Jul-80 13:09:18
```

```
 FLOPER.EXE;2010 34304(36) 18-Jul-80 13:58:47

 FLTEST.SAV;2000 25088(36) 20-Jun-80 10:19:18
```

... A Sample Change Notice

```
;CONFIG.INDEX;2 & CONFIG.WORK;56 9-Jun-80 0811    PAGE 2


LINE 35, PAGE 1
1)     FLPKG.SAV;25500 46592(36) 11-Mar-80 06:56:56
1)
LINE 35, PAGE 1
2)     FLPKG.SAV:25700 46592(36) 22-May-80 05:05:47
2)

LINE 41, PAGE 1
1)     FOREMAN.SAV;1610 26112(36) 10-Mar-80 12:20:35
1)
LINE 41, PAGE 1
2)     FOREMAN.SAV;1612 26112(36) 3-Jun-80 03:35:37
2)

LINE 49, PAGE 1
1)     WM.SAV;29300  120832(36) 11-Mar-80 07:12:15
1)
LINE 49, PAGE 1
2)     WM.SAV;29500  120832(36) 23-May-80 12:50:09
2)
```

Our original goal was CM auditing which could be performed by a Data Technician, with other personnel involved only if changes were found. If differences like the previous example are found, they may be forwarded to

configuration managers for disposition; if FILCOM displays "NO LINES CHANGED" the CM audit has terminated with the watchdog silent.

6.3. Software Trouble Reports

If limitations are to be known and removed over time, each problem must be centrally reported and new ones carried in a journal for the life of the product. Consider the needs of different organizations:

- User - Where can I report my problem? How can I work around it? When will it be fixed?
- Developer - Are there any problems with my pieces? Here is a component which fixes problems a and b.
- Managers - Which problems will be addressed by the next system release? What is the workload for each of the developers? Rank the outstanding problems by severity.

As contrasted with the CM watchdog who seldom barks, STR processing requires direct participation by nearly every project person. Each STR will be in a different state according to its progress and the subset of project staff affected.

Clearly, no existing tool could meet such needs, so MONSTR was specially designed and built - based largely on the Works Manager's database system, but operating over a computer-based problem journal instead of NSW's database of names, permissions, and file catalogue entries. This system is used by project personnel in all categories to report and deal with

limitations.

6.4. Testing

The large number of interacting components found in a network operating system dictates a requirement for extensive testing. Each of these components must be unit tested, of course, and the set of components on a single host can be tested by usual methods. Once each host can function locally, the combinatorics of integration testing immediately arise. For example, the combination of hosts involved when an interactive tool needs to get a (possibly remote) NSW file is roughly cubic in the number of hosts in the network: FE*FM*FP, where the three variables are respectively the number of hosts offering Front Ends, Foreman, and File Packages.

Our approach towards dealing with such a large number of test scripts is random testing by our Data Technician on a daily basis. Using NSW tools, we have written a program which - using a random number generator and tables of probabilities - chooses a different sequence of host locations and tool choices on each execution of the generator. (As the random seed is the time since midnight in hundredths of a second, duplicate runs are acceptably impossible.) The table of probabilities and two resulting test scripts are shown below. These scripts are then run by the Data Technician, but given availability of a testing tool, the script could be run automatically.

TEST CONFIGURATION IS AS FOLLOWS:

HOST NAME AND CHOICE PROBABILITY

USC-ISIC	18
USC-ISIE	18
UCLA-CCN	22
MULTICS	22
RADC-20	20

HOST NAME OFFERING TOOL NAME AND PROBABILITY

USC-ISIC	SOS-IC	15
USC-ISIE	SOS-IE	15
UCLA-CCN	FTN-UC	25
MULTICS	QEDXRM	30
RADC-20	SOS-R2	15

HOST NAME OFFERING FTP AND AND PROBABILITY

USC-ISIC	FTP-IC	30
USC-ISIE	FTP-IE	30
RADC-20	FTP-R2	40

HOST NAME OFFERING FRONT END AND PROBABILITY

USC-ISIC	30
USC-ISIE	35
RADC-20	35

```
  GENERATE TEST SCRIPT WITH 20 STEPS

LOG INTO NSW USING FE AT RADC-20
 USE FTP-IC TO SEND FILE TO USC-ISIE
 TRANSPORT FILE FROM USC-ISIE TO UCLA-CCN
 USE FTP-R2 TO GET FILE FROM UCLA-CCN
 EXPORT FILE TO UCLA-CCN
 TRANSPORT FILE FROM UCLA-CCN TO RADC-20
LOG OUT OF FRONT-END AT RADC-20
 - - - - - - - - - - - - - - - - -
LOGIN TO FRONT-END AT USC-ISIE
 IMPORT FILE AT RADC-20
 EXPORT FILE TO UCLA-CCN
 IMPORT FILE AT UCLA-CCN
 USE FTP-IC TO SEND FILE TO UCLA-CCN
 USE FTP-R2 TO GET FILE FROM UCLA-CCN
LOG OUT OF FRONT-END AT USC-ISIE
 - - - - - - - - - - - - - - - - -
LOGIN TO FRONT-END AT RADC-20
 USE FTP-IE TO SEND FILE TO MULTICS
 TRANSPORT FILE FROM MULTICS  TO UCLA-CCN
LOG OUT OF FRONT-END AT RADC-20
 - - - - - - - - - - - - - - - - -
LOGIN TO FRONT-END AT USC-ISIE
 IMPORT FILE AT UCLA-CCN
 EXPORT FILE TO UCLA-CCN
 USE FTP-R2 TO GET FILE FROM UCLA-CCN
 USE QEDXRM TO MAKE ANOTHER NSW FILE COPY
 EXPORT FILE TO MULTICS
LOG OUT OF FRONT-END AT USC-ISIE
 - - - - - - - - - - - - - - - - -
LOGIN TO FRONT-END AT RADC-20
 IMPORT FILE AT MULTICS
LOG OUT OF FRONT-END AT RADC-20
 - - - - - - - - - - - - - - - - -
LOGIN TO FRONT-END AT USC-ISIE
 USE FTP-R2 TO SEND FILE TO USC-ISIC
LOG OUT OF FRONT-END AT USC-ISIE
 - - - - - - - - - - - - - - - - -
LOGIN TO FRONT-END AT RADC-20
 USE FTP-IE TO GET FILE FROM USC-ISIC

   SCRIPT IS NOW COMPLETE
```

```
  GENERATE TEST SCRIPT WITH 20 STEPS

LOG INTO NSW USING FE AT RADC-20
 EXPORT FILE TO MULTICS
LOG OUT OF FRONT-END AT RADC-20
 - - - - - - - - - - - - - - - - -
LOGIN TO FRONT-END AT USC-ISIC
 USE FTP-IE TO GET FILE FROM MULTICS
LOG OUT OF FRONT-END AT USC-ISIC
 - - - - - - - - - - - - - - - - -
LOGIN TO FRONT-END AT RADC-20
 EXPORT FILE TO UCLA-CCN
 USE FTP-IC TO GET FILE FROM UCLA-CCN
 USE FTN-UC TO MAKE ANOTHER NSW FILE COPY
LOG OUT OF FRONT-END AT RADC-20
 - - - - - - - - - - - - - - - - -
LOGIN TO FRONT-END AT USC-ISIE
 USE SOS-IE TO MAKE ANOTHER NSW FILE COPY
LOG OUT OF FRONT-END AT USC-ISIE
 - - - - - - - - - - - - - - - - -
LOGIN TO FRONT-END AT RADC-20
 EXPORT FILE TO USC-ISIC
 USE FTP-IE TO GET FILE FROM USC-ISIC
 EXPORT FILE TO UCLA-CCN
LOG OUT OF FRONT-END AT RADC-20
 - - - - - - - - - - - - - - - - -
LOGIN TO FRONT-END AT USC-ISIC
 TRANSPORT FILE FROM UCLA-CCN TO UCLA-CCN
 USE FTP-IE TO GET FILE FROM UCLA-CCN
LOG OUT OF FRONT-END AT USC-ISIC
 - - - - - - - - - - - - - - - - -
LOGIN TO FRONT-END AT USC-ISIE
 USE QEDXRM TO MAKE ANOTHER NSW FILE COPY
 USE FTN-UC TO MAKE ANOTHER NSW FILE COPY
 USE SOS-R2 TO MAKE ANOTHER NSW FILE COPY
 EXPORT FILE TO USC-ISIC
 IMPORT FILE AT USC-ISIC
 USE QEDXRM TO MAKE ANOTHER NSW FILE COPY
 USE SOS-IE TO MAKE ANOTHER NSW FILE COPY
LOG OUT OF FRONT-END AT USC-ISIE
 - - - - - - - - - - - - - - - - -
LOGIN TO FRONT-END AT RADC-20
 USE QEDXRM TO MAKE ANOTHER NSW FILE COPY
LOG OUT OF FRONT-END AT RADC-20
 - - - - - - - - - - - - - - - - -
```

```
LOGIN TO FRONT-END AT USC-ISIE
 EXPORT FILE TO RADC-20

   SCRIPT IS NOW COMPLETE
```

CHAPTER 7

FUTURE DIRECTIONS: CONVERSION TO TCP/IP

In order to continue to operate NSW in the future (i.e. 1983) the conversion of NSW to use the TCP/IP networking protocols is perhaps the most challenging: it will require the modification of many separate programs, on all of the host types which take part in NSW.

The following discussion centers on the the TOPS-20 changes as a basis for explaining the factors which must be considered. At the time of writing this document, we have in hand all the program sources for all TOPS-20 NSW components, as well as the details of the TCP/user-program interface which will be implemented in TOPS-20; hence we can speak with some assurance about that system. We do have as detailed information on the corresponding UNIX systems; the factors for consideration are of course the same as those explained below for TOPS-20, but the details of implementation will, of course, be different.

It is especially to be noted that we will not have a TCP-in-UNIX test-bed available as soon as we will have a TOPS-20 TCP installation to use. It is for this reason that our proposed program for the implementation and testing of the TCP conversion leaves the UNIX components to the last.

7.1. Background

MSG is the interprocess communication facility for the National Software Works (NSW). NSW is implemented as a number of processes running concurrently on a number of different computer systems, called hosts. The hosts are connected by a communications network; this is currently the ARPA Network (ARPANET). However, the MSG facility is designed to be as independent as possible of the ARPANET implementation so that the concepts may be carried over to implementations on other networks.

Although it was designed specifically to satisfy NSW communication requirements, MSG is a general purpose inter-host interprocess communications facility which is useful for other applications. The SDD1 database system (of Computer Corporation of America) employs MSG.

In the NSW context, an MSG program must exist on each host of the network where one or more of these NSW processes runs:

- Front End - Works Manager - File Package - Foreman (and some number of tools).

Each of these processes communicates to another such process via MSG. MSG supports communication between such processes both within a host and across hosts; the network is not used for intra-host communication.

Although the principal purpose of MSG is communication, it also performs some rudimentary process control functions, such as process allocation. MSG implementations also provide comprehensive facilities for monitoring, controlling, and debugging processes under MSG control.

The MSG System/Subsystem Specification is host independent; there are implementations for:

- PDP-10 TENEX - DECsystem-20 TOPS-20 - IBM 3033 MVS and TSO
- Honeywell 6180 Multics - PDP-11/70 BBN UNIX

MSGs currently communicate with one another using the ARPANET Host-Host Protocol (AHHP), which is part of the ARPANET Network Control Protocol (NCP). The protocols employed by MSGs are layered upon AHHP.

As computer networking has been playing an increasingly important role in military, government, and civilian sectors, it has become essential to provide communication among hosts of various networks; i.e., networks are becoming interconnected. Thus the need has arisen for adopting a standard interprocess communication protocol which can be used to provide a reliable service in a multinetwork environment. DoD has declared that Transmission Control Protocol (TCP) is to be adopted by all hosts on DoD packet-switching networks by January 1983.

Therefore, since AAHP availability is being retired, all MSGs must be modified to call upon TCP facilities instead. COMPASS proposes to coordinate this effort among all appropriate participants and furthermore to be specifically responsible for the TOPS-20 and UNIX implementations of MSG.

7.2. More about TCP

TCP is a connection-oriented, end-to-end protocol designed to fit into a layered hierarchy of protocols. TCP assumes it can obtain a simple, potentially unreliable datagram service from lower-level protocols. Just below TCP is the Internet Protocol (IP) -- also part of the DoD standardization. IP provides the means by which TCP can send and receive variable length segments of information packaged in internet datagram "envelopes". The datagram provides for addressing source and destination hosts (by fixed length addresses). IP also provides for fragmentation and reassembly of long datagrams, if necessary, for transmission through "small packet" networks. Since MSG services are built upon the TCP layer, we will not dwell on the IP facilities.

The interface between an application process and TCP consists of a set of calls much like the calls an operating system provides for file manipulation. For example, there are calls to open and close connections and to send and receive data on established connections. The interface to

TCP also includes a provision for asynchronous communication (e.g. traps). These are the same sorts of functions which have been provided to applications which employ AHHP.

7.2.1. TCP/AHHP Differences

TCP provides approximately the same functions as AHHP, with the addition of strong end-to-end error control. In particular, TCP provides full duplex connections whose ends are labeled with 16-bit numbers called ports. In AAHP each end of a connection is identified by a host number and a 32-bit socket number, and AAHP connections are uni-directional. Unlike AHHP, TCP allows the same port (on a given host) to participate in any number of connections whose remote ends have differing <host, port> pairs.

At a level above AHHP, an Initial Connection Protocol (ICP) has been employed to provide a convenient standard method for several processes (such as MSGs) to gain simultaneous access to some specific process (such as MSG) at another host. ICP is neither required nor supported along with TCP/IP.

TCP messages consist of 8-bit bytes, called octets, whereas AAHP has permitted a variety of connection byte sizes.

A TCP segment may include a field called the "Urgent pointer" which

indicates there is "urgent" data a specified number of bytes ahead in the data stream. Although the Urgent pointer is outside the data stream proper, it is not exactly like the "interrupt" control messages of AAHP; however, TCP's Urgent pointer can be used to achieve the same function as the AHHP interrupt in many contexts.

7.2.2. AAHP to TCP Transition

Because all ARPANET hosts could not be converted to provide TCP support simultaneously, there has been a transition plan which has been followed over the past couple of years. Until the transition period is over, some hosts will support only NCP, some will support only TCP, and others will support both (simultaneously). Hosts in the latter category can serve as "relay hosts" for Telnet, FTP, and Mail services.

The fact that an ARPANET host may support both protocols simultaneously permits the development of TCP-based MSGs while AHHP-based MSGs continue to be used for production purposes.

7.2.3. TCP Support on TOPS-20 and TENEX

Bolt, Beranek & Newmann, Inc. (BBN) has provided a package for user programs to call upon TCP facilities via JSYSes (monitor calls). This has been in use at some West Coast ARPANET sites which are running Release 4 of TOPS-20. This package will not be employed in TOPS-20 Release 5; instead,

Digital Equipment Corporation (DEC) is currently producing a JSYS interface. Unfortunately, the BBN package is not consistent with the "TOPS-20 way of doing things"; in particular, JFNs are not employed by BBN. (BBN uses an indexable handle for a connection, called a Job Connection Number or JCN which can only be used as an argument to other JSYSes of the TCP package.)

It is COMPASS's concern that MSG modification will be based on a freshly-created package which has not been "shaken down" during months of active use. We assert that reliable TCP support should be in place at RADC TOPS-20 by the end of October 1982 so that we may produce a reliable TCP-based MSG by January 1983.

7.3. More about MSG

MSG provides three different modes for communicating among NSW processes: messages, alarms, and direct communication paths. Message exchange is the most common mode of communication; to send a message, a process addresses it either to some specific other process or generically to any of a class of processes. Alarms provide a means for one process to alert another to the occurrence of an exceptional or unusual event; this mode of communication is typically satisfied by interrupts in other interprocess communication systems. A direct communication path can be made (by requests from a pair of processes) to support high volume or very frequent

communication, such as file transfers between hosts or communication between an NSW Front End and a Foreman supporting a tool.

The MSG facility implements these modes of communication by opening ARPANET connections when appropriate. A pair of communicating MSGs sustains an open connection until such time as there is no activity between them. A host/host connection is also used to realize a direct communication path.

All modes of interprocess communication supported by MSG follow the same basic pattern, which is roughly as follows:

- One process tells MSG about a message or alarm to be sent or a connection to be opened. It also specifies a destination address and a signal by which MSG can inform it that the message or alarm has been sent or the connection opened.
- Another process which matches the above destination address tells MSG that it is ready to receive the same type of communication. It also specifies a signal by which MSG can inform this process that the message or alarm has been received or the connection opened.
- MSG sends the alarm or message or opens the connection. It also signals the source process that the message or alarm has been sent or the connection opened and signals the destination process that the message or alarm has been delivered or the connection opened. After it receives the signal, the process receiving a message or alarm always knows the specific address of the sender.

7.3.1. TENEX vs TOPS-20 MSG

At present there is one executable core image file which will run under either the TENEX or TOPS-20 operating systems. The original implementation was for TENEX, and it was subsequently modified to allow it to run under TOPS-20 as well.

In certain situations MSG must execute operating system dependent code because the two operating systems are not identical. The principal differences between TENEX and TOPS-20 that impact MSG relate to the interprocess communication facility and the JSYS trap mechanism. On TENEX the signal (SID) facility must be used for inter-job interprocess communication. Because TOPS-20 does not support TENEX-like signals, another mechanism, called IPCF, must be used in its place. The call and return conventions for the JSYS trap mechanism differ between the two systems, and so use of the mechanism by MSG is operating system dependent. There are a number of other minor areas where operating system dependent code must be executed.

We do not necessarily intend to continue to support a TENEX MSG when MSG is modified for TCP use, but it will likely turn out that the program will continue to be usable under that operating system. There are almost no TENEXes on the ARPANET and they will soon be retired; at least we do not expect NSW to be used on them.

For the remainder of this document only "TOPS-20" MSG will be mentioned.

7.3.2. TOPS-20 MSG

An MSG configuration for a TOPS-20 host is implemented by a collection of time-sharing jobs. There are two different types of jobs used in a configuration: one single "central" MSG and some number (possibly zero) of "process controlling" MSG jobs.

The central MSG job is responsible for system initialization and communication with MSG implementations on other hosts. In addition, it is responsible for the allocation of certain resources shared among the MSG jobs, such as mutual exclusion semaphores, and for timing out pending events associated with process operations.

The process controlling jobs directly control communicating user processes. These jobs interact with one another and the central MSG as necessary to support process communication.

The central MSG job consists of a number of different processes or TOPS-20 "forks"; there are at least five forks, where one additional fork (namely PTCL) is required for each open MSG-to-MSG connection. Two of the other five forks (namely ICPSER and SGLSER) along with the PTCL forks implement the MSG-to-MSG protocol, and this is where MSG must be modified

to work with TCP instead of AHHP (via the TOPS-20 Network Control Program - NCP).

The TOPS-20 MSG program is implemented in MACRO-10, the assembly language for the PDP-10 (or DECsystem-10) and equivalent processors. Modification and maintenance of MSG requires understanding of the TOPS-20, MACRO-10, MSG, and the ARPANET. The COMPASS technical staff are uniquely prepared to design and carry out the necessary modifications to MSG, second only, perhaps, to certain BBN staff members. We understand that the knowledgeable BBN staff are not available to carry out these tasks during the required time period.

At our disposal, in addition to commented assembly listings and a Program Maintenance Manual, we have established working relationships with key personnel at BBN. As part of this effort, we intend to be able to call upon BBN staff as needed for some small amount of consultation relating to this task.

7.4. NSWSTA — MSG Subtool

There is an MSG subtool named NSWSTA (for NSW STAtus) which currently runs on the TOPS-20 at RADC. As directed by a command file (basically a list of hosts), NSWSTA wakes up every so often (usually at about 20 minute intervals) and determines which of the NSW hosts are up in the sense that

their MSGs are alive and responding.

NSWSTA performs an ICP to the MSG reserved socket at each appropriate host, but as soon as the receiver seems willing to communicate, NSWSTA closes the connection. As a result of these probes a text file is produced which in turn is observable by TOPS-20 users via the INFO NSW EXEC command at RADC-TOPS20.

As MSG is modified to operate with TCP, the NSWSTA program must correspondingly be changed. Although this is a relatively small task, it is nevertheless another step required for the migration.

7.5. Work to be Done

In order for NSW to continue to operate on the ARPANET, all MSG programs must be converted to communicate via TCP. COMPASS proposes to coordinate the required modifications to the MSG Specification and specifically to carry out program modifications on the TOPS-20 and UNIX implementations of MSG. As part of these tasks we will update the appropriate documentation and take over the program and document maintenance functions for the TOPS-20 and UNIX MSG.

7.5.1. MSG-to-MSG Protocol

The first step in migrating from AHHP to TCP is the establishment of agreed-upon host-independent changes to the MSG-to-MSG protocols. We do not envision any significant changes here with respect to representation of the 25 MSG-to-MSG protocol commands, except for the CONN-OPEN command. Currently, the arguments of CONN-OPEN are in terms of AHHP notions, specifically: 32-bit socket numbers, uni-directional connections, and size of the connection (in bits).

The other major area for required change is in how MSGs open (and close) connections to one another. We believe this will be a simplification in MSG organization since ICP will no longer be required. Namely, connections in TCP can share port numbers at a host, and MSG on each host can employ its reserved port number to sustain its end of each of the connections it has to other MSGs. This implies the MSGs can be modified to use full duplex connections. If this does not prove to be acceptable to all participants, a pair of port numbers may have to be reserved at each MSG host and half-duplex connections will continue to be used.

Although TCP/IP can support the sending of a datagram without the overhead of opening (and closing) a host-host connection, the basic architecture of all MSG servers is based on retained connections. It would be a mistake to modify MSG architecture to make use of the datagram

service.

The only way in which there can now be more than one MSG-to-MSG connection between a given pair of NSW hosts is as a result of each host of the pair initiating a connection to the other at approximately the same time. The TCP design accounts for this case and results in only one connection.

The MSG-to-MSG protocol for closing MSG-to-MSG connections must be redesigned in the light of TCP.

These and perhaps other isssues will be discussed by all organizations who are responsible for the various MSGs. COMPASS will be responsible for updating the MSG System/Subsystem Specification.

7.5.2. Alternate NSW Configurations

To support NSW as a production product there have grown to be four independent systems which can be in use simultaneously; these are named user, candidate, development, and debug. The purpose of having these different systems is for proper quality control of the production "user" system. Associated with each active NSW system on a host is its own MSG; i.e. each MSG serves the components of a single NSW system. It is necessary to keep the MSGs on one host from interfering with one another,

specifically from using the same socket numbers. This detail was solved administratively within AHHP/ICP by assigning to the user system a "well known" socket number, and for the other systems, keeping tables of socket numbers for all hosts in the system. On TOPS-20 hosts, these are "directory relative" socket numbers. At present, the plan for TCP support on TOPS-20 does not include directory-relative ports, as such, but reserves a range of numbers for "user assigned" ports, which can probably be used in the same fashion.

Since NCP and TCP/IP traffic is handled separately there can be both old and new MSGs on a host which supports both protocols.

7.5.3. 8-bit Connections

TCP will support only 8-bit connections. This presents no problem for MSG-to-MSG communication, since that is already specified and implemented in terms of messages of 8-bit bytes. There may be problems with the use of the CONN-OPEN command by some of the processes of NSW; in particular, we are concerned about NSW File Packages opening 36-bit connections via MSG. It will be necessary to perform some form of minor modifications to the TOPS-20 File Packages as part of supporting NSW under TCP; the TOPS-20 File Package uses 36-bit connections to talk to another TOPS-20, but the standard File Package uses 8-bit connections across hosts.

7.5.4. Host Classification

Currently, network facilities under NCP include a classification of hosts by host type. The NSW File Package makes use of this classification to determine whether to make a "family copy" or a file translation; this facility will continue to be needed under TCP. This is an area to be investigated.

7.5.5. Alarms

Another area to investigate is the implementation of MSG alarms within the TCP domain. The "urgent data" facility of TCP simulates an "out of band" signalling mechanism for TCP traffic, much as does the MSG Alarm facility with respect to normal MSG messages; it is not clear, however, that it would not just be a complication of MSG logic flow to accept an urgent-data interrupt (where provided) from its networking substrate.

7.5.6. TOPS-20 MSG Specifics

Our plan is to modify the TOPS-20 MSG program so that it uses TCP primitives instead of AHHP (NCP) primitives. The resulting new version will work only with TCP. There would be unwarranted complexity in producing some hybrid version which could deal with both old and new network protocols.

The TOPS-20 MSG program will be modified in line with agreements

established for MSG-to-MSG protocols discussed above. In addition there are some changes to be made in the user (application)/MSG interface. One area which needs some consideration is the 16-bit host address; TCP/IP host numbers are 32-bit quantities, but it is not necessary that the user (application) use these numbers. Some mapping will have to be done -- we do not consider it worth the effort to change the MSG calling sequences at this time. Too much operational code would be affected.

A small item which must change is the meaning of the DBGSW internal switch which controls several aspects of MSG operation.

There are two text files used to define MSG configurations: the generic name file and the network configuration file. Host specifications in both files must be redefined in the presence of TCP, and socket specification in the network configuration file must be redefined.

The contents of internal data structures of TOPS-20 MSG will be affected by the migration to TCP. Specifically, the Connection Control Block (CCB) must be redefined.

The TOPS-20 MSG User Manual will be updated as part of this activity.

7.5.7. Quality Assurance for MSG

COMPASS plans on making the first changes to MSG on TOPS-20 only, since that is a well understood environment with the best debugging tools for network development. First, we will be sure MSG continues to operate properly within one host. Next, we would have two TOPS-20 sites communicate with one another, and then we could add other TOPS-20 hosts. Testing MSG-to-MSG communication can be done using two standard MSG measurement and test processes called M1 and M2. A possible alternate approach is to make a small patch in MSG to force it to use the network yet call upon another MSG on the same host. Debugging could then proceed using PTYCON, a multi-job controlling facility in TOPS-20.

Once MSGs communicate via M1 and M2, the next step is to bring up an NSW. However, recall that some of the other NSW components will have to be converted.

The n TOPS-20 communicate with each host individually and then have them communicate with one another. After these hosts are operational, we propose to bring up the UNIX MSG.

7.5.8. NSW Dispatcher

Another component of NSW which requires modification in light of the AHHP to TCP transition is the Dispatcher. This is a relatively small program for the TOPS-20 which is an adjunct to the TOPS-20 NSW Front End. The Dispatcher provides remote terminal access to a Front End via a special ICP contact socket. This provides a facility for a TELNET user (e.g. from a TIP) to gain "direct" access to NSW rather than logging in to an NSW host; early during the NSW session the user is required to supply an NSW password.

The Dispatcher is currently contacted by an AHHP ICP, and it creates a job which includes a process controlling MSG and a Front End. In converting to TCP we expect a simplification in assignment of sockets (or ports), as has been described for MSG-to-MSG connections. For each of the four alternate NSW configurations there must be a port number reserved on each TOPS-20 which suspports Dispatcher service.

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.

END

FILMED

6-83

DTIC